US005689589A

United States Patent [19]

Gormish et al.

[11] Patent Number: 5,689,589

[45] Date of Patent: Nov. 18, 1997

[54] **DATA COMPRESSION FOR PALETTIZED VIDEO IMAGES**

[75] Inventors: **Michael J. Gormish**, Los Altos; **Martin P. Boliek**, San Francisco, both of Calif.

[73] Assignees: **Ricoh Company Ltd.**, Tokyo, Japan; **Ricoh Corporation**, Menlo Park, Calif.

[21] Appl. No.: **347,789**

[22] Filed: **Dec. 1, 1994**

[51] **Int. Cl.**[6] **G06K 9/46**; H04N 1/40

[52] **U.S. Cl.** **382/239**; 382/232; 382/236; 382/233; 348/415; 341/107

[58] **Field of Search** 382/236, 232, 382/233, 235, 239, 248, 251, 253; 341/107, 51, 73, 76; 348/416, 415; 358/426

[56] **References Cited**

U.S. PATENT DOCUMENTS

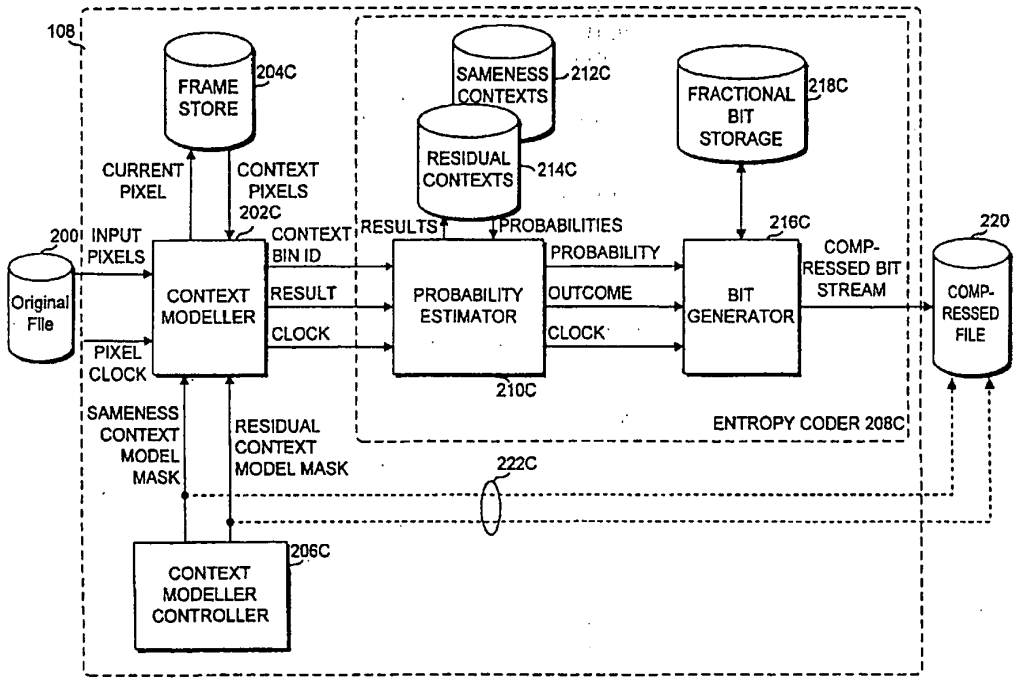
4,717,956	1/1988	Moorhead et al.	348/416
5,298,992	3/1994	Pietras et al.	348/415
5,313,204	5/1994	Semasa et al.	341/107
5,414,423	5/1995	Pennebaker	341/107
5,422,724	6/1995	Kang	382/232
5,442,458	8/1995	Rabbani et al.	358/426
5,471,207	11/1995	Zandi et al.	382/232
5,550,540	8/1996	Furlan et al.	341/51

Primary Examiner—Leo Boudreau
Assistant Examiner—Wenpeng Chen
Attorney, Agent, or Firm—Philip H. Albert; Townsend and Townsend and Crew LLP

[57] **ABSTRACT**

A data compression system uses sameness information, such as temporal sameness of corresponding pixels, in the coding process. Two sets of contexts are used, one set when a pixel is the same as a sameness pixel, and one set of contexts for residual coding of the pixel when it is not the same. The use of the sameness bit saves computation because, if in decoding the one "sameness" bit, a decompressor determines that the pixel is equal to the corresponding pixel in a previous frame, then no further decoding is needed for that pixel.

23 Claims, 7 Drawing Sheets



Docket no. 2134

EXHIBIT A

EXHIBIT A

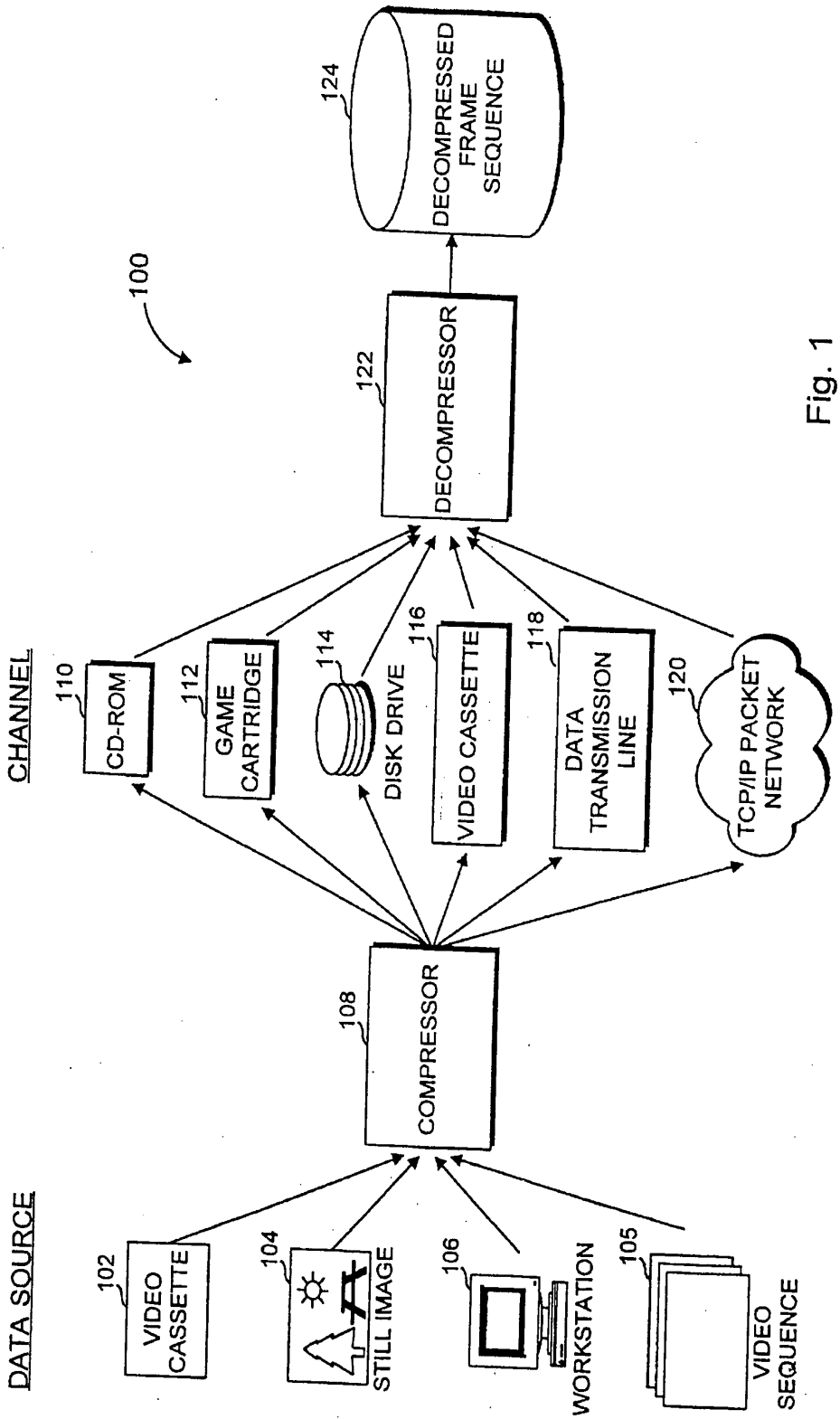

100
DECOMPRESSED FRAME SEQUENCE
124
DECOMPRESSOR
122
CHANNEL
CD-ROM
110
GAME CARTRIDGE
112
DISK DRIVE
114
VIDEO CASSETTE
116
DATA TRANSMISSION LINE
118
TCP/IP PACKET NETWORK
120
COMPRESSOR
108
DATA SOURCE
VIDEO CASSETTE
102
STILL IMAGE
104
WORKSTATION
106
VIDEO SEQUENCE
105

Fig. 1

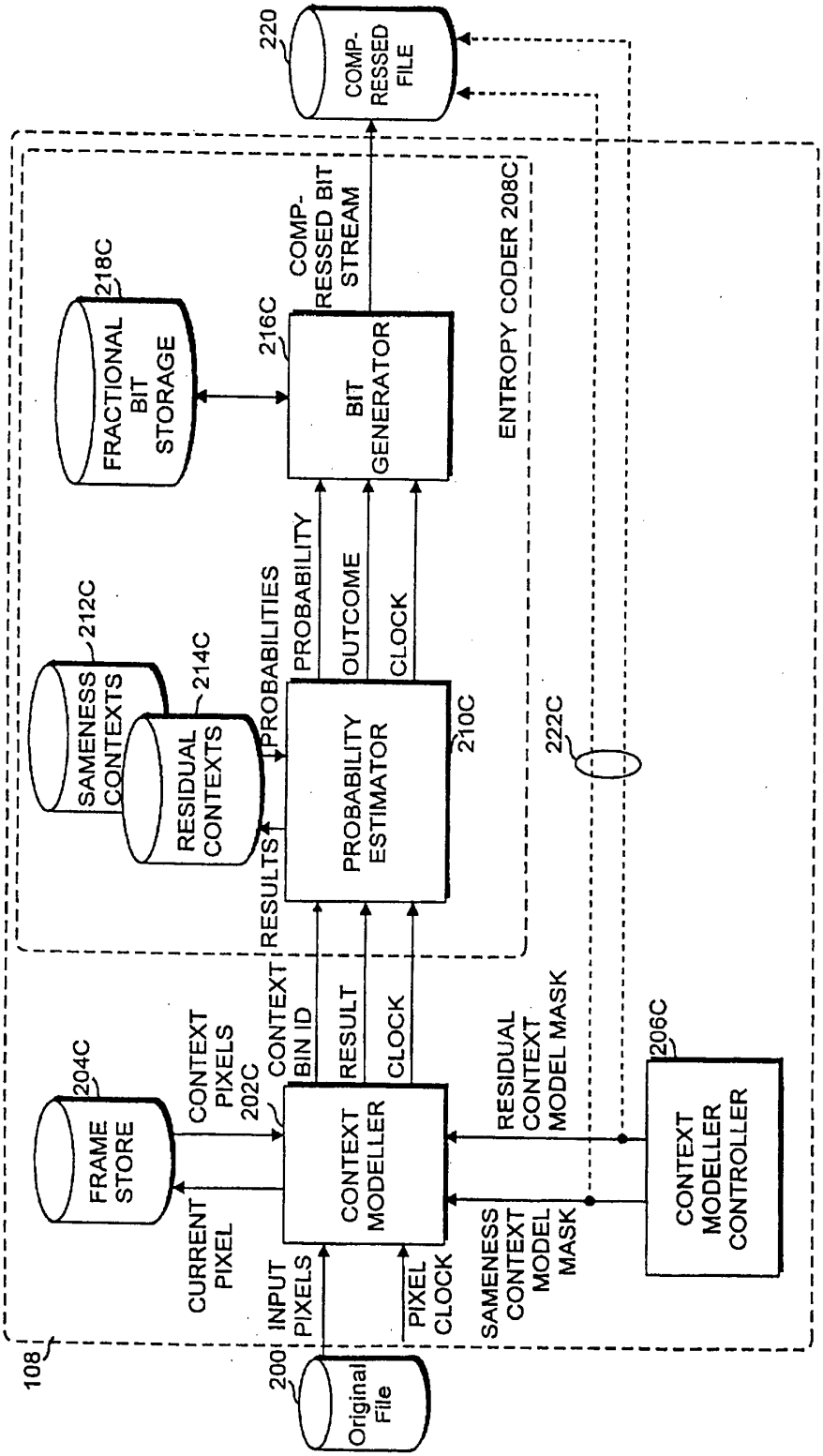

220
COMP-RESSED FILE
ENTROPY CODER 208C
COMP-RESSED BIT STREAM
218C
FRACTIONAL BIT STORAGE
216C
BIT GENERATOR
212C
SAMENESS CONTEXTS
214C
RESIDUAL CONTEXTS
PROBABILITIES
PROBABILITY
OUTCOME
CLOCK
PROBABILITY ESTIMATOR
210C
222C
RESULTS
CONTEXT BIN ID
RESULT
CLOCK
RESIDUAL CONTEXT MODEL MASK
206C
204C
FRAME STORE
CONTEXT PIXELS
202C
CONTEXT MODELLER
CONTEXT MODELLER CONTROLLER
CURRENT PIXEL
INPUT PIXELS
PIXEL CLOCK
SAMENESS CONTEXT MODEL MASK
108
200
Original File

Fig. 2

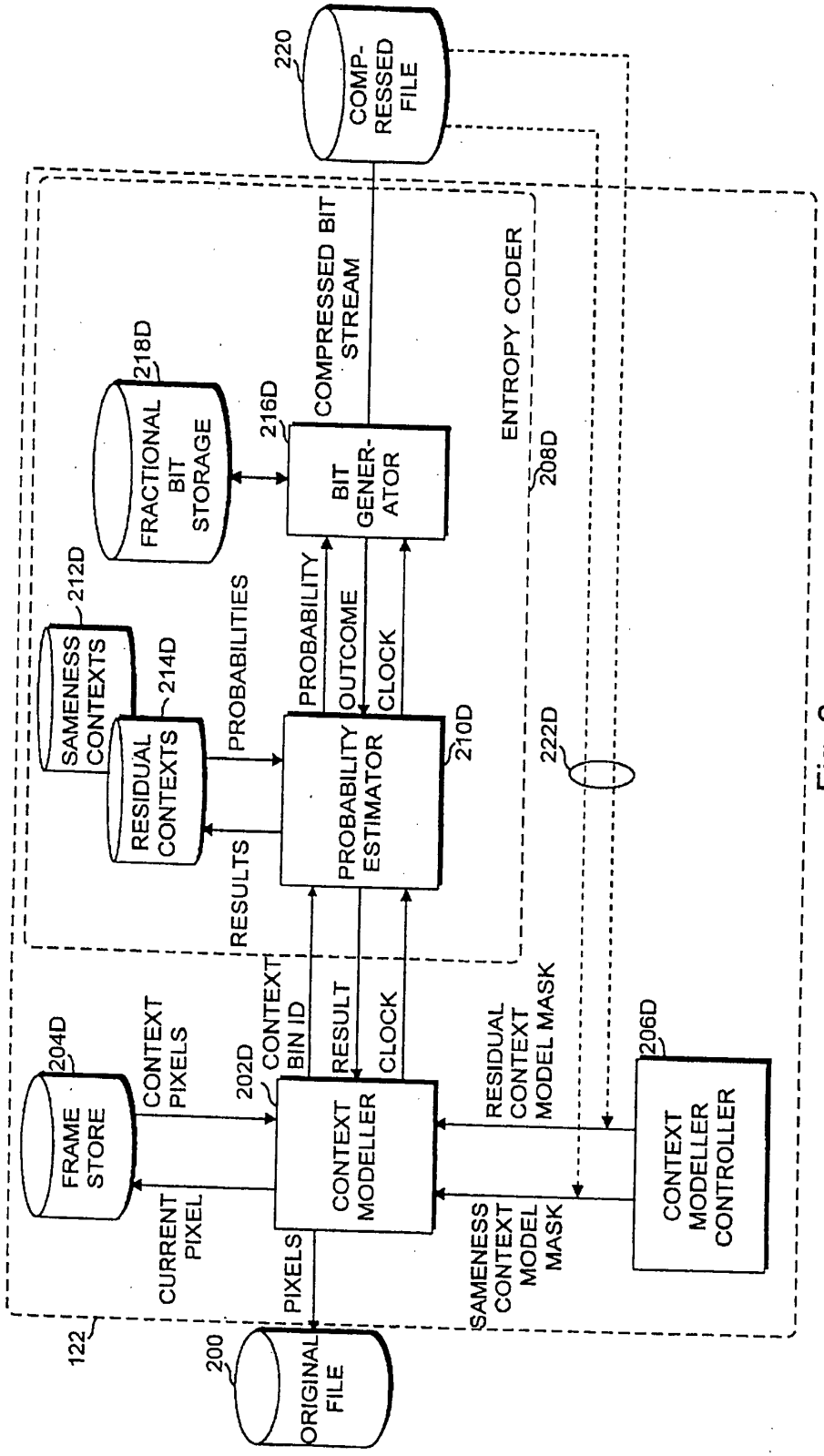

220
COMP-RESSED FILE
COMPRESSED BIT STREAM
ENTROPY CODER
208D
218D
FRACTIONAL BIT STORAGE
216D
BIT GENER-ATOR
212D
SAMENESS CONTEXTS
214D
RESIDUAL CONTEXTS
PROBABILITIES
PROBABILITY
OUTCOME
CLOCK
210D
PROBABILITY ESTIMATOR
222D
RESULTS
CONTEXT BIN ID
RESULT
CLOCK
204D
FRAME STORE
CONTEXT PIXELS
202D
CONTEXT MODELLER
RESIDUAL CONTEXT MODEL MASK
206D
CONTEXT MODELLER CONTROLLER
CURRENT PIXEL
PIXELS
SAMENESS CONTEXT MODEL MASK
122
200
ORIGINAL FILE

Fig. 3

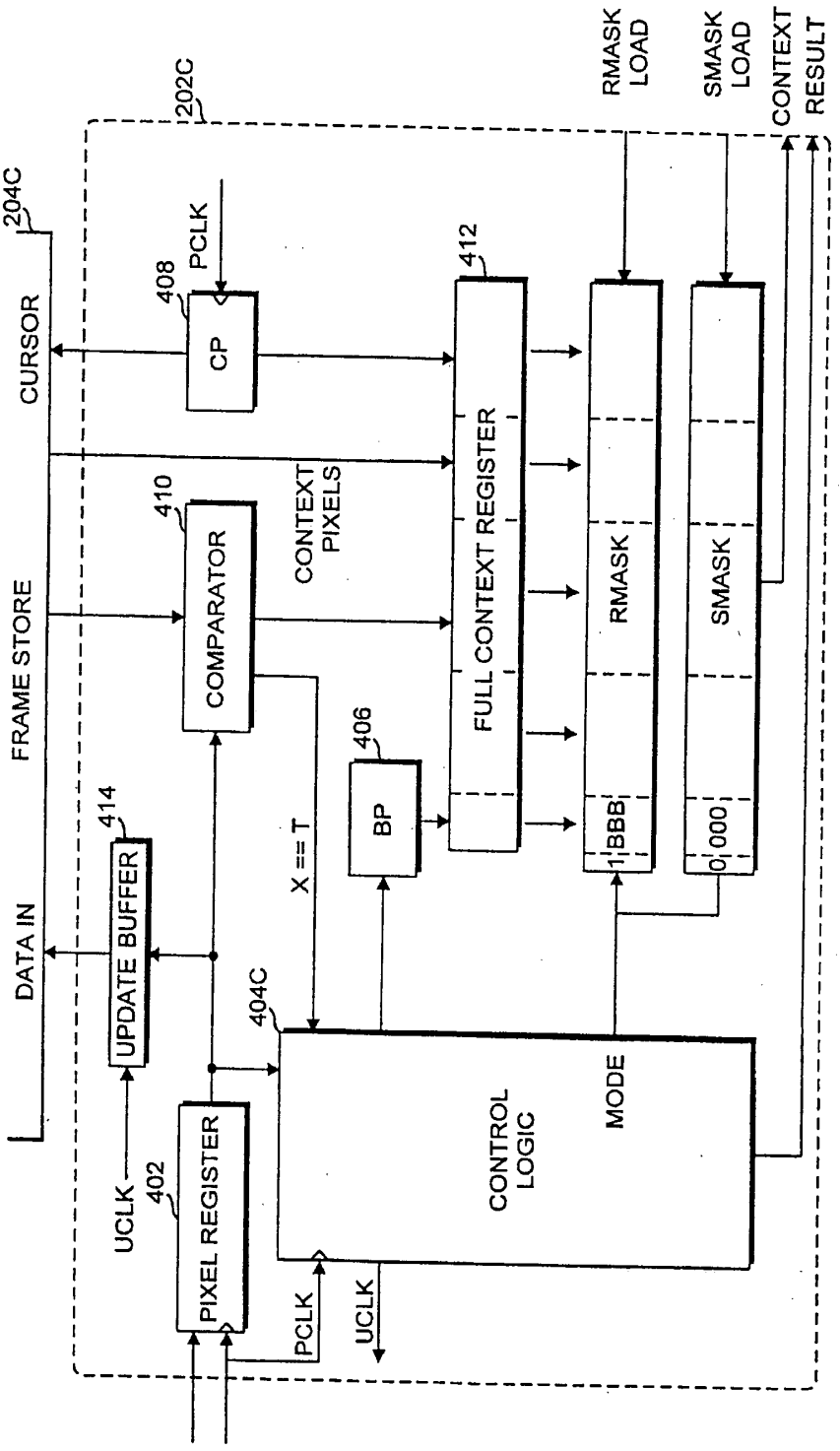


Fig. 4

Docket no. 2134

EXHIBIT A

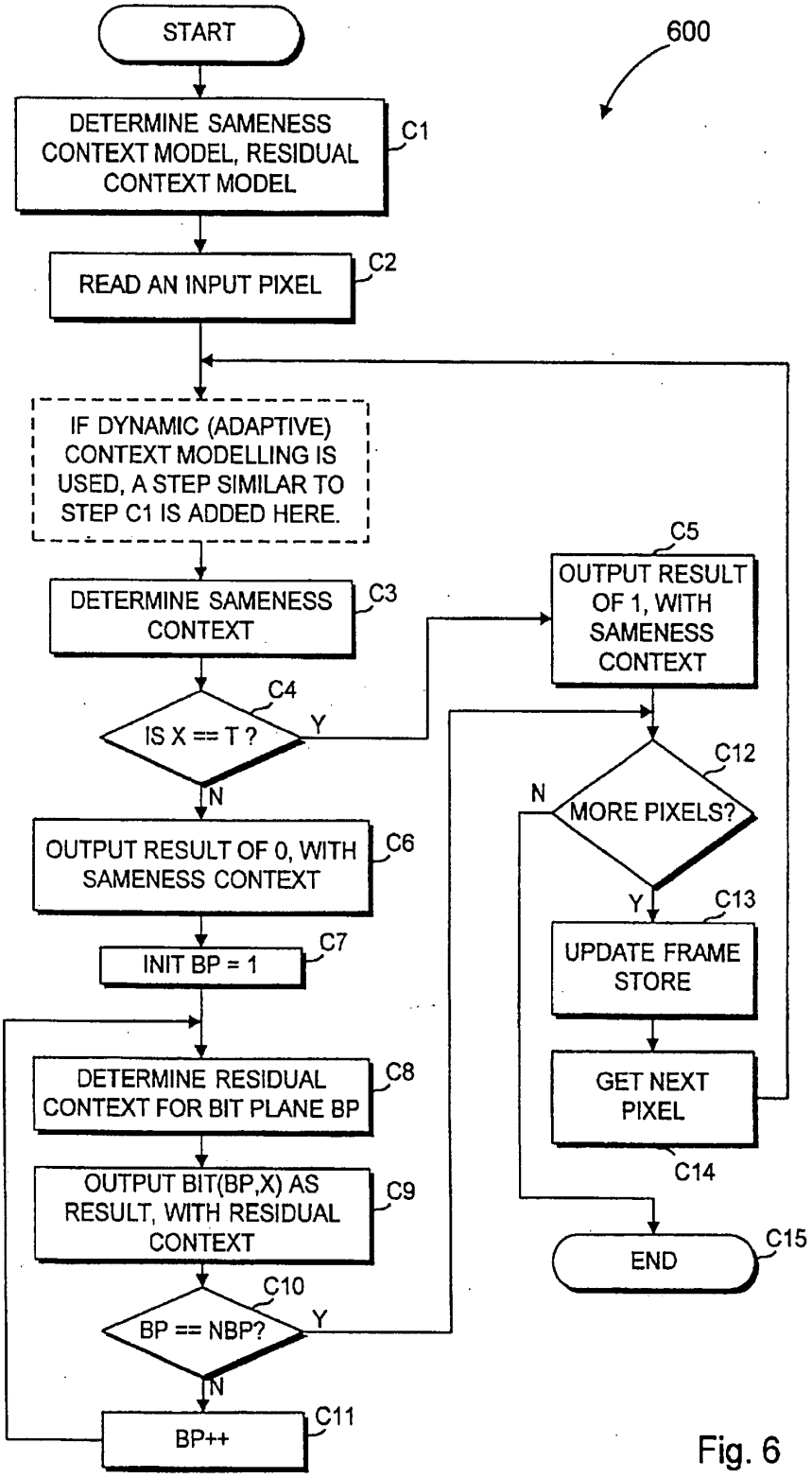

600
START
DETERMINE SAMENESS CONTEXT MODEL, RESIDUAL CONTEXT MODEL
C1
READ AN INPUT PIXEL
C2
IF DYNAMIC (ADAPTIVE) CONTEXT MODELLING IS USED, A STEP SIMILAR TO STEP C1 IS ADDED HERE.
DETERMINE SAMENESS CONTEXT
C3
IS X == T ?
C4
Y
N
OUTPUT RESULT OF 1, WITH SAMENESS CONTEXT
C5
OUTPUT RESULT OF 0, WITH SAMENESS CONTEXT
C6
INIT BP = 1
C7
DETERMINE RESIDUAL CONTEXT FOR BIT PLANE BP
C8
OUTPUT BIT(BP,X) AS RESULT, WITH RESIDUAL CONTEXT
C9
BP == NBP?
C10
Y
N
BP++
C11
MORE PIXELS?
C12
N
Y
UPDATE FRAME STORE
C13
GET NEXT PIXEL
C14
END
C15

Fig. 6

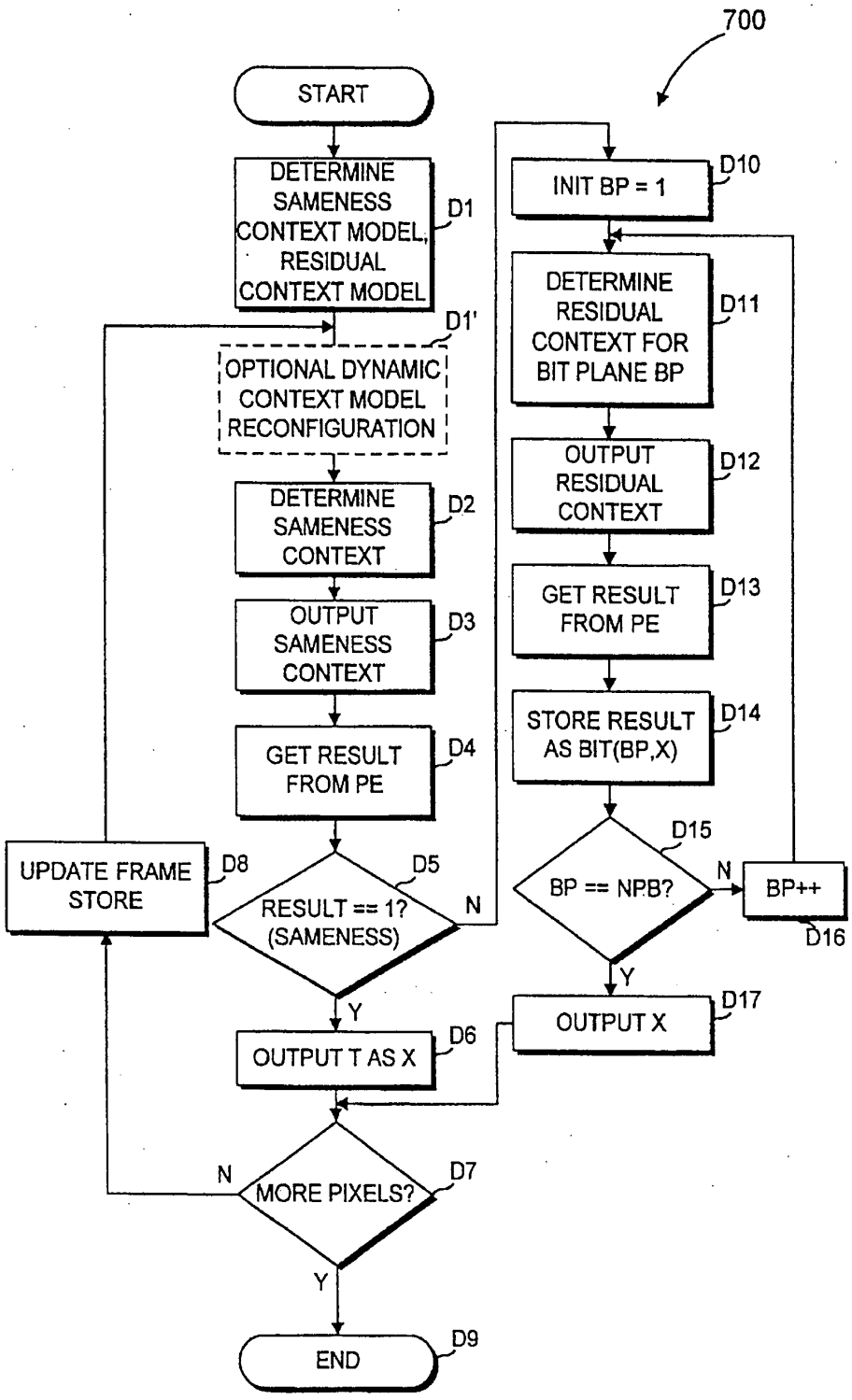

700
START
DETERMINE SAMENESS CONTEXT MODEL, RESIDUAL CONTEXT MODEL
D1
D1'
OPTIONAL DYNAMIC CONTEXT MODEL RECONFIGURATION
DETERMINE SAMENESS CONTEXT
D2
OUTPUT SAMENESS CONTEXT
D3
GET RESULT FROM PE
D4
RESULT == 1? (SAMENESS)
D5
N
Y
UPDATE FRAME STORE
D8
OUTPUT T AS X
D6
MORE PIXELS?
D7
N
Y
END
D9
INIT BP = 1
D10
DETERMINE RESIDUAL CONTEXT FOR BIT PLANE BP
D11
OUTPUT RESIDUAL CONTEXT
D12
GET RESULT FROM PE
D13
STORE RESULT AS BIT(BP,X)
D14
BP == NPB?
D15
N
BP++
D16
Y
OUTPUT X
D17

Fig. 7

DATA COMPRESSION FOR PALETTIZED VIDEO IMAGES

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of several applications using compression and decompression to efficiently store or transmit data;

FIG. 2 is a more detailed view of the compressor shown in FIG. 1;

FIG. 3 is a more detailed view of the decompressor shown in FIG. 1;

FIG. 4 is a more detailed view of the context modeller of the compressor shown in FIG. 2;

FIG. 6 is a flow chart of the precoding process performed by a context modeller; and

FIG. 7 is a flow chart of the decoding process performed by a context modeller.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

FIG. 1 is a block diagram of a generalized application where compression is used.

FIG. 1 shows several data sources, such as a video cassette **102**, a still image **104**, a video sequence **105** and a workstation **106** which produces rendered images. Where the data source is image data, and especially sequences of image frames, compression is needed because of the large amount of memory needed for uncompressed images or sequences. The present invention is described with reference to frame sequences, however the present invention is useful with other types of data sharing similar characteristics of frame sequence data.

The data is input to a compressor **108** which, if designed correctly, outputs a compressed image or images containing fewer bits than the original data. If compressor **108** is a lossless compressor, the original data is exactly recoverable from the bits of the compressed data. Once the data is compressed, it is applied to a channel, several examples of which are shown in FIG. 1.

FIG. 2 shows compressor 108 in greater detail, along with an original file 200 representing a sequence of uncompressed images (frames) and a compressed file 220 representing a compressed version of original file 200.

Compressor 108 moves the original data through a context modeller 202C, and an entropy coder 208C to form compressed file 220. Context modeller 202C has an input for accepting original pixels and a pixel clock.

Context modeller 202C is also coupled to a frame store 204C, to update frame store 204C with the current pixel and to read context pixels or context bits from frame store 204C.

In operation, during each cycle of the pixel clock, compressor 108 accepts an input pixel (the "current" pixel) from file 200, which contains the original pixels. In response to the current pixel, compressor 108 outputs a number of bits to compressed file 220, where the number depends on the state of compressor 108, and can be zero bits (i.e. no bits output in a given input pixel cycle). In the zero bit output case, the internal state of 108 is changed. This will be reflected in a later compressed bit or bits.

Context modeller 202C accepts the input pixel and outputs one or more sets each of a context bin ID and a result value.

Context modeller 202C determines the context bin ID and result, according to methods explained in more detail below.

Initially frame store 204C is empty, so it might not be able to provide contexts for very early pixels. However, the frame store could just be filled with an a priori value so that contexts can be provided for every result. After the first frame, frame store 204C contains at least a full frame. frame store 204C fills, the oldest pixels are overwritten as new pixels are stored. Frame store 204C need only be large enough to hold the values needed to determine future contexts. For example, if no context referred to pixel values from earlier than the immediately prior frame, storage is only needed for one full frame.

FIG. 3 shows decompressor 122 in more detail, having many of the same elements as compressor 108. To distinguish similar elements of decompressor 122 and compressor 108, the elements of decompressor 122 are indicated by similar numbers but with a "D" suffix instead of a "C" suffix, such as context modeller 202D, frame store 204D, context model controller 206D and entropy decoder 208D comprising probability estimator 210D, context tables 212D. 214D. bit generator 216D and fractional bit storage 218D.

The update clock clocks update buffer **414** to update frame store **204C** after the context for the current pixel is obtained. Of course, if frame store **204C** has enough space to hold the current pixel without overwriting any pixels which form the context for the current pixel, then PCLK can be substituted for UCLK.

BP register **406**, comparator **410**, frame store **204C**, and CP register **408** each provide bits or pixels to full context register **412**.

The operation of context modeller **202C** during one pixel clock (PCLK) cycle will now be described. First, the current pixel is read into pixel register **402**. CP register **408** "addresses" frame store **204C** to obtain the context pixels for the current pixel. An exemplary context model is one where the pixel to the left of the current pixel forms the context. In this case, the pixel to the left would be read out of frame store **204C**, with CP register **408** indicating where the current pixel is, and thus where the "pixel to the left" is to be found. Some of the context pixels are compared by comparator **410** to one another and to the current pixel. As shown in FIG. 4, the current pixel (designated "X") is compared to the pixel in the immediately previous frame which occupies the same pixel position as the current pixel does in the current frame (designated "T"). The result of this comparison is provided to control logic **404C**. The results of other comparisons, such as the pixel to the left compared with the corresponding pixel from a previous frame, are provided to full context register **412**.

FIG. **6** is a flow chart of the process followed by context-modeller **202C** to convert the current pixel into a result and a context for that result. First, context model controller **206C** determines the sameness context model and the residual context model and loads SMASK and RMASK (step C1). Once those are set, control logic **404C** cycles the pixel clock to load the current pixel into pixel register **402** (step C2). In some embodiments, control logic **404C** adaptively changes the masks in a causal manner.

Next, the sameness context is determined (step C3) by updating full context register **412** and masking with SMASK. If dynamic (adaptive) context modelling is used, the dynamic context model is determined first (step C3'). With the current pixel being X, the sameness test X=T is done (C4), and if that test result is true, a result value of 1 is output (C5) along with the sameness context of the current pixel. The sameness test tests whether or not the current pixel is the same as the corresponding pixel from a previous frame (i.e., whether the color of the current pixel changes from the previous frame). This test is represented by X=T and is performed by comparator **410**, which reads T from frame store **204C** and X from pixel register **402**. If X=T is true, no other information about the current pixel needs to be output.

After the residual bit output, the bit plane is compared to the number of bit planes (N BP), to check if it is the last bit plane (C10). If more bit planes remain, the BP register is incremented (C11), and the process repeats at step C8. Otherwise, the process continues at step C12, where a test is done to check if any more pixels remain to be compressed. This step also follows step C5. If more pixels remain to be processed, control logic 404C cycles UCLK to update frame store 204C (C13) and then cycles PCLK to get the next pixel (C14). If no more pixels remain, the process terminates (C15).

FIG. 7 is a flow chart of the decompression process which is described with reference to FIG. 3 as well as FIG. 7. This process is performed by decompressor 122, including context modeller 202D. Context modeller 202D is similar to context modeller 202C, except that context modeller 202D inputs a result and outputs an input, thus requiting different control logic.

Context modeller 202D checks the result (D5). If it is 1, indicating that the current pixel X was the same as T, context modeller 202D reads T from frame store 204D and outputs T as the current pixel (D6). Context modeller 202D then checks for more pixels (D7). If there are more pixels, frame store 204D is updated with pixel X (D8), otherwise the process ends (D9).

* * * * *